Rotor: Bringing .NET to FreeBSD

Stephen R. Walli
Program Manager, Rotor Project

Outline

- Project Goals
- The License
- The ECMA standards work
- Rotor architecture
- Platform adaptation layer
- Base Class Library Selection

What is Rotor?

- Rotor is a reference implementation of the ECMA CLI and C# specs, plus key SDK tools, delivered as source under a license with generous terms for non-commercial purposes.
- It will build and run on Windows and FreeBSD

Why Are We Building Rotor?

- Key reason: support of proposed ECMA standard
  - Proof point for alternate platforms
  - Approachable "guide" for teachers, experimenters, and other commercial implementers
- ECMA standard relates compact framework, .NET framework, and third-party implementations
- Why bootstrap alternate CLI implementations?
  - Establish new runtime platform around web services
  - Interop will be a key to customer satisfaction
  - "Interop" is not "cross platform" (WORA is non-goal)

Shared Source CLI License

- Part of Microsoft's shared source framework
  - Multiple licenses offer flexibility and choice
  - Maintains the intellectual property rights needed to support strong software ecosystem
  - Adopts beneficial aspects of open source models, especially in the areas of source transparency and community
- One representative license will be for the shared source CLI, C#, and ECMAScript implementations:
  - Neither published nor final, but similar to published WinCE license
  - Feedback welcome!
- Goals of license design
  - Long-term investment of intellectual property into intellectual commons
  - Very liberal rights w.r.t. non-commercial derivatives, to increase educational access to seed the future
  - "Taintless" viewing of source when used as an implementation guide

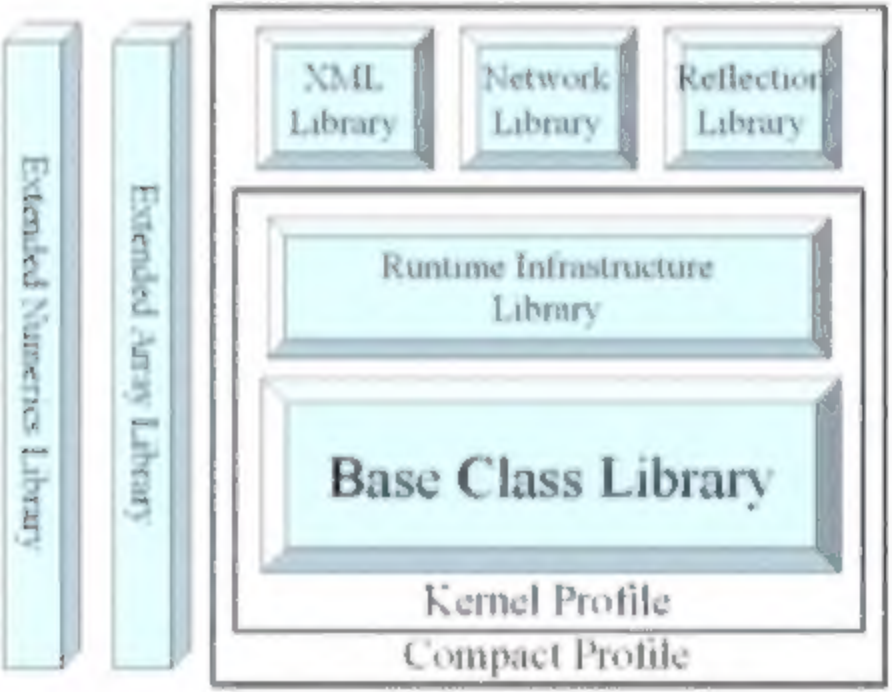
ECMA Submissions

- Partition I: Concepts and Architecture
  - Includes common type system
- Partition II: Metadata Definition and Semantics
  - Includes ilasm syntax, file format, PE and metadata
- Partition III: CIL Instruction Set
  - Includes instruction validation and verification rules
- Partition IV: Profiles and Libraries
  - Describes partitioning and common material
- Partition V: Annexes
  - Validation, verification, design guidelines, etc.
- C# Programming Language
  - Complete language specification, CLI is runtime for C#

Detail: Base Class Libraries

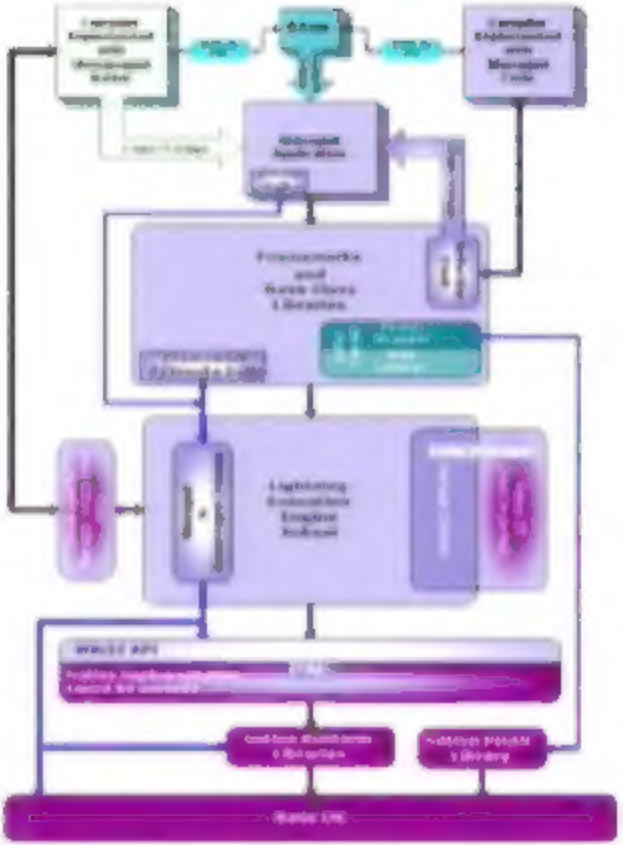
- Abstract OS Interface
  - Networking, Security, Standard I/O, Threading
- Common Programming Library
  - Common Data Types, Advanced Data Types, Collections, Extended Numerics, Regular Expressions, Serialization
- High-Level Programming
  - Asynchronous Programming, Globalization, Remoting, XML, Advanced XML
- EE Functionality
  - GC, Hosting, NonCLS, Reflection, Unmanaged
- Miscellaneous
  - Kernel, Basic Language Support, Development Time

Map of Libraries within Profiles



Interesting Rotor Components

- PAL
- Alternate JIT
- Simpler GC
- Hosting and calls into managed code
- Platform Independent GUI



PAL

- PAL implements platform support in a generic way
  - Boot loader
  - Threading support
  - Networking
  - Timers
  - File system
- PAL APIs preserve Win32 aspect but are chosen to facilitate porting to Unix platforms
  - Designed to minimize changes in DNoW code base
- PAL used by the
  - CLR
  - FX classes via P/Invoke
  - Unmanaged tools
- About 200 Win32 APIs will be implemented
  - Eliminating WinForms simplified it considerably

PIGUI Plan

- Many competing [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
- Many competing [illegible] libraries
  - [illegible]
    - [illegible] GTK+ vs KDE on top of Qt
    - [illegible]
  - [illegible]
    - [illegible]
    - Qt has licensing issues on Windows
    - GTK+ has ugly programming model and no Win32 option
    - Th[illegible] look and feel and [illegible] slow
- [illegible]
  - Fits with ubiquity goal
  - [illegible]

Rules for Cutting

- Started with everything in DNoW and then cut
  - Didn't build up from ECMA standards
- Cut criteria:
  - Remove trade secret or other sensitive IP
    - In most cases there are alternate low IP code bases
  - Remove Windows-specific functionality that can't feasibly be ported in v1
  - Remove features that are beyond the basics

Significant Cuts and Exceptions

- **COM interop cut**
  - [illegible] target platforms
  - Expensive cut [illegible]
- **Windows Forms cut**
  - Too dependent on [illegible] platform [illegible]
- **ASP.NET cut**
  - But System.Web.Services in R[illegible]
- **MS-specific areas of data classes cut**
  - [illegible]L server and OLE DB providers are cut
  - Other areas under investigation
- **VS.NET support cut**
  - No cross-dev scenario
  - [illegible]
  - All Designer classes cut
  - VB and [illegible] compiler will not be ported
  - But C# and JScript will be ported
- **Transparent loading and IJW support**
  - Rotor will not load [illegible] images
  - Users must use [illegible] to execute managed apps

Availability

- Q1/Q2 2002 for beta release
- Q3/Q4 2002 for final release
- Community build exercise around the release

Additional Resources

- Project Web site:
  http://rotor
- ECMA specs:
  http://msdn.microsoft.com/net/ecma
- Whitepaper and FAQ:
  http://www.microsoft.com/net/whitepapers.asp
- Microsoft commercial C# and CLR beta:
  http://msdn.microsoft.com/downloads
- Shared source info
  http://www.microsoft.com/sharedsource
  http://www.microsoft.com/windows/embedded/ce

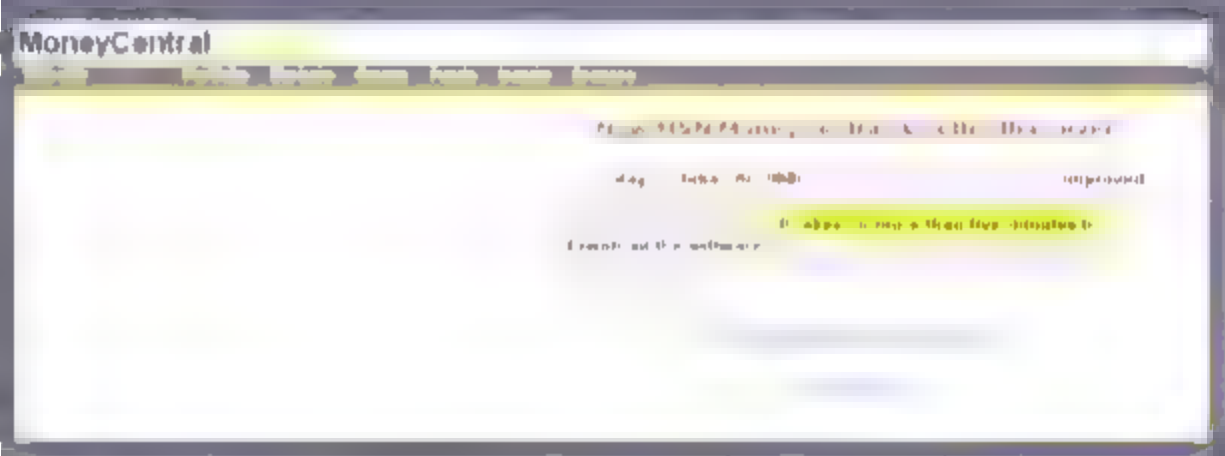
Questions?

Loader NET
Software On Demand

Hoi Vo

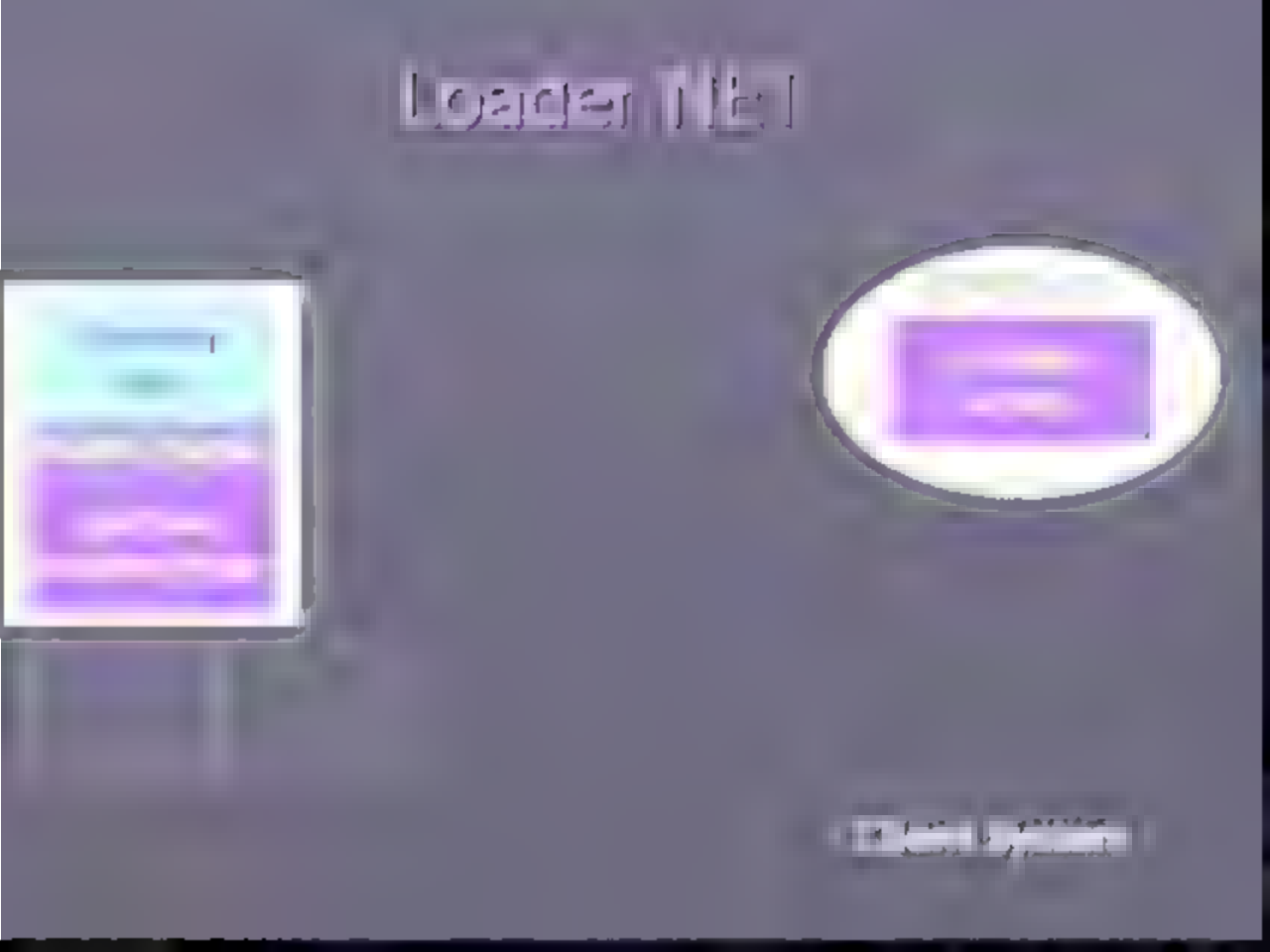
Agenda

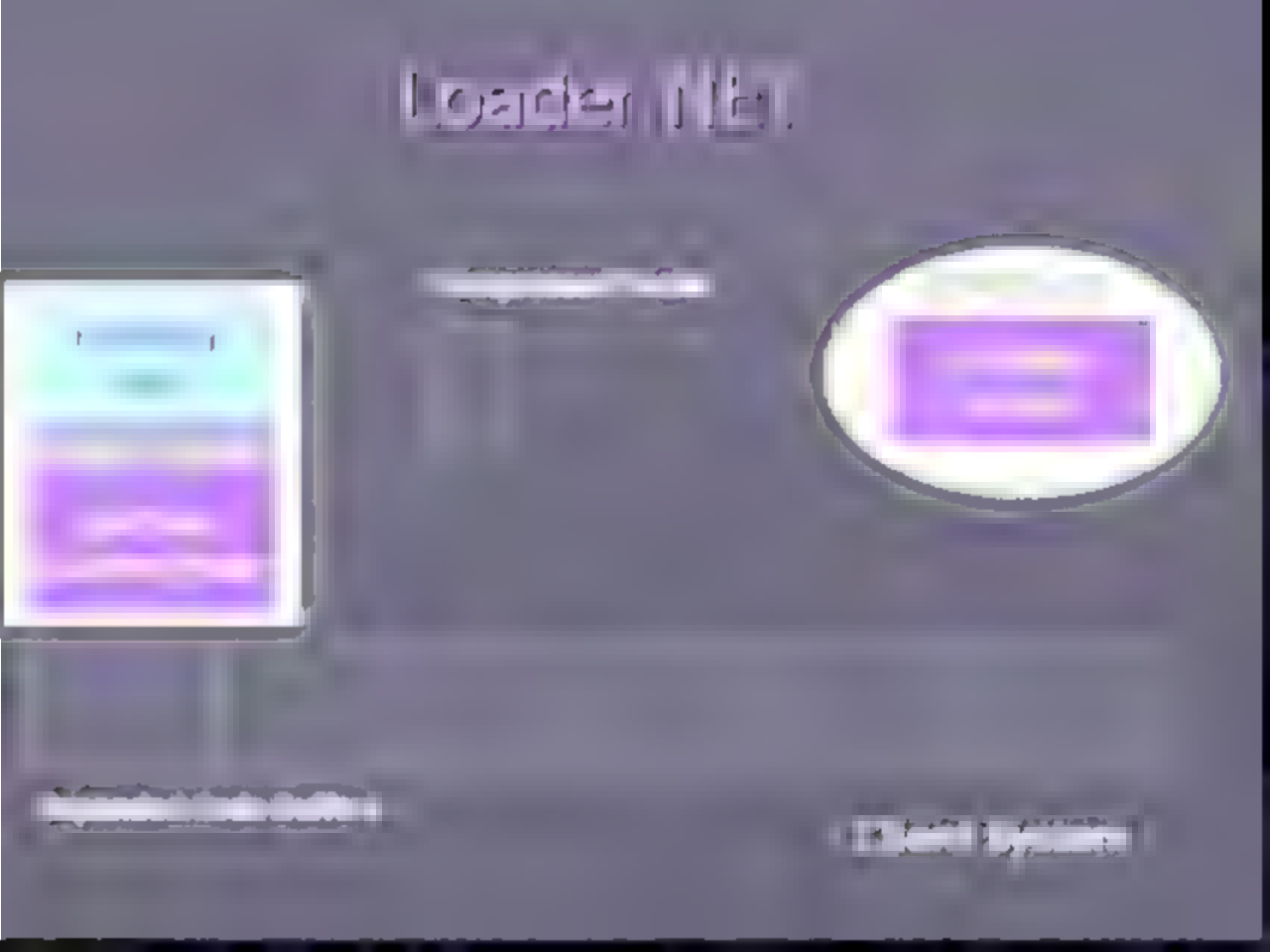
Introduction

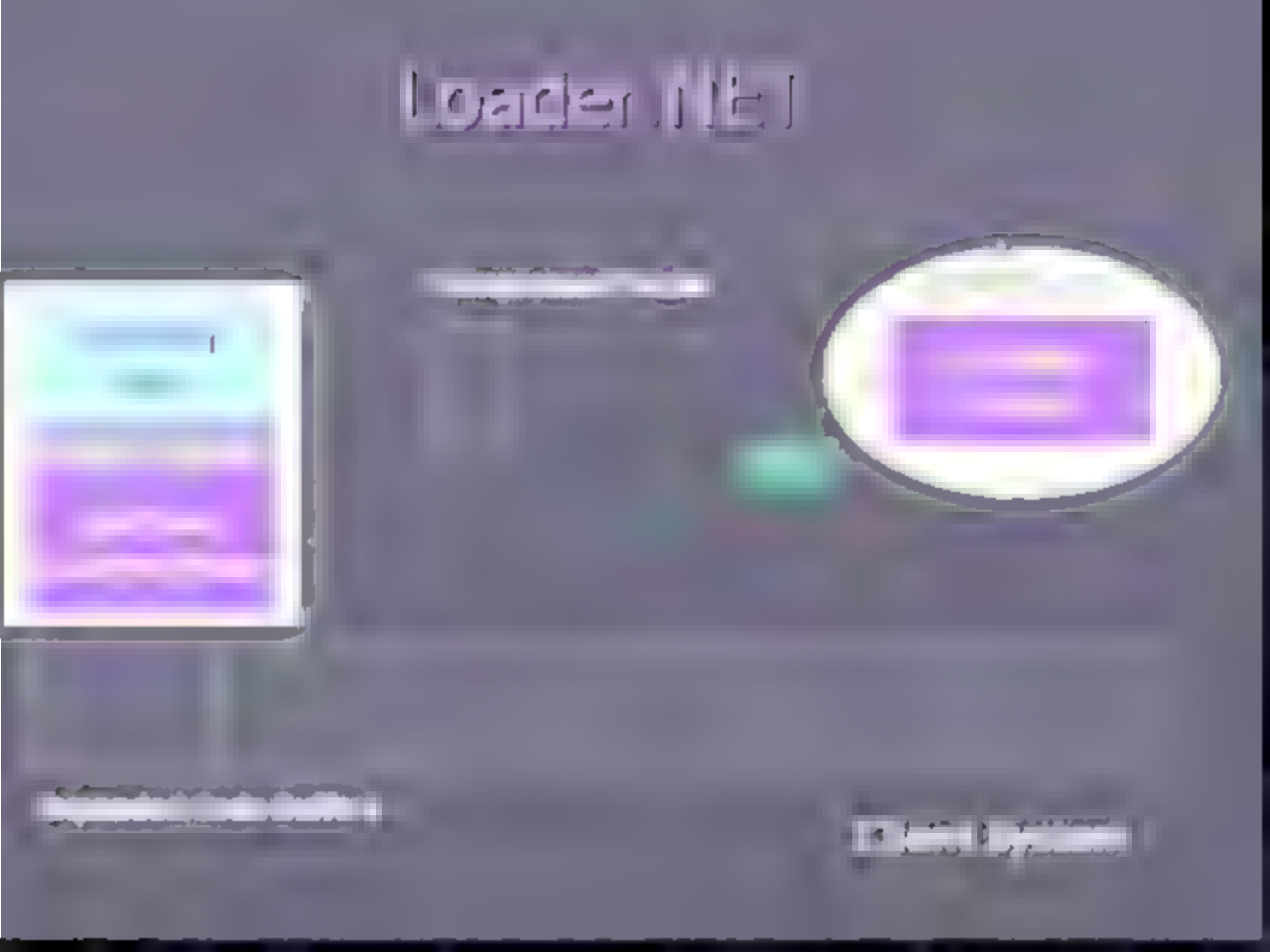


[illegible] challenges

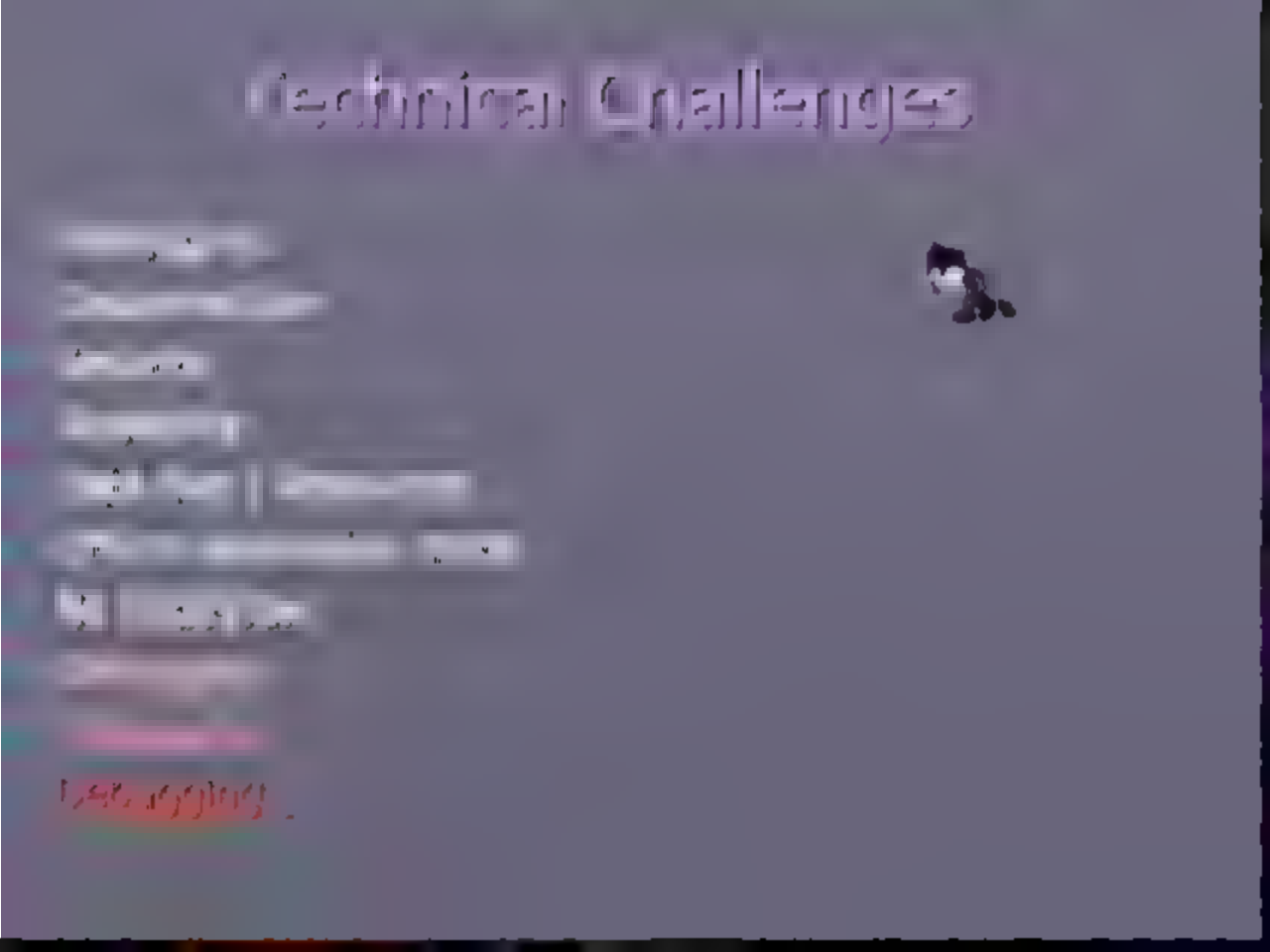
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]
- [illegible]


Loader Net




Loader.NET

Technical Challenges



Sparse Binary Connect

Sparse Binary Coding

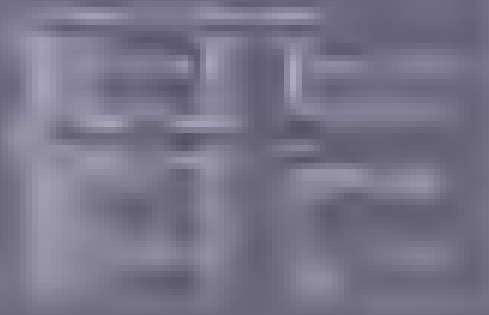
problem

compression ratio

Sparse Binary Coding

problem

compression ratio



Sparse Binary Coded

problem

compression ratio

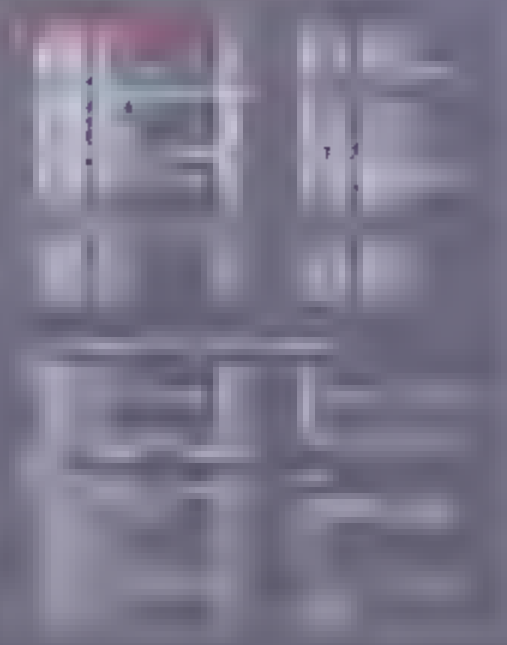
upgrade

Sparse Binary Coreset

problem

compression ratio

upgrade

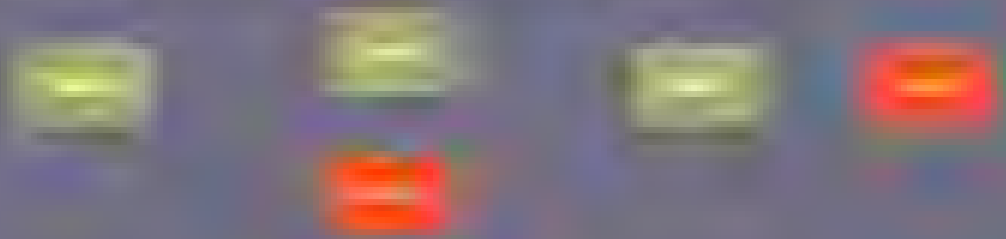


Handling Data Blocks

- Issues:
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]

Handling Data Blocks

- Issues:
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]
  - [illegible]



Handling Data Blocks

- Issues:
  - Multiple references from different code paths
  - R/W data blocks require stricter sync during decomp
  - Too large data arrays require special handling
  - Special PE data blocks cannot be streamed
  - Singly referenced data blocks detection

Handling Data Blocks

- Issues:
  - Multiple references from different code paths
  - R/W data blocks require stricter sync during decomp
  - Too large data arrays require special handling
  - Special PE data blocks cannot be streamed
  - Singly referenced data blocks detection

Download Cache Optimization

- Use BBT profile data for static layout
- Cached funclet layout schemes:
  - Layout based on funclet loading order
  - Layout based on BBT layout order
  - Layout based on pre-computed order
- One download cache for each Loader.NET component
- Multiple processes share same download cache for the same component

Into the Future

- MSIL streaming support
  - Metadata per class
  - CLR hook for class loading and release
  - Pre-JIT binary support
- Integration into platform (i.e. Fusion)
- Offline application support
  - Drizzle mode support
- Universal download cache